# Transient signal flagging algorithm definition for radiance data

**Summary:** This document provides the description of transient signal flagging algorithm for radiance data to be implemented in the GDPS. The function is described with the associated input and output data.
.

| | | Date | Signature |
|---|---|---|---|
| **Author:** | Q. L. Kleipool | July 12, 2005 | |
| **Checked:** | N. Rozemeijer | | |
| **Approved:** | M. Dobber | | |
| **Archive:** | R. Noordhoek | | |

## DISTRIBUTION LIST

| | |
|---|---|
| Pieternel Levelt | KNMI |
| Marcel Dobber | KNMI |
| Bert van den Oord | KNMI |
| Nico Rozemeijer | TriOpSys |
| M. ter Linden | Dutch Space |
| V. Schenkelaars | Dutch Space |
| T. Watts | Dutch Space |
| Harry Förster | NIVR |

## CHANGE STATUS

| Issue | Date | Author | Comments | Affected pages |
|---|---|---|---|---|
| 1 | June 13, 2005 | Q. L. Kleipool | Initial Version | All |
| 2 | July 12, 2005 | Q. L. Kleipool | modified tables for the OPF | 8 |

## References

- [rd_01] Wirth, Niklaus, *Algorithms + Data structures = Programs*, **p. 84**. Prentice Hall, 1976, ISBN: 0130224189

## Acronyms

| | | |
|---|---|---|
| CCD | : | Charge Coupled Device |
| OMI | : | Ozone Monitoring Instrument |
| OPF | : | Operational Parameter File |
| SAA | : | Southern Atlantic Anomaly |
| SNR | : | Signal to Noise Ratio |
| UV | : | Ultra Violet |
| VIS | : | Visual |

# 1 Transient signal flagging algorithm for radiance data

It has been observed that OMI is not perfectly shielded from radiation events in its current space environment. Radiation that impinges on the two OMI CCD devices introduce transient signals (i.e. spikes) in the observations. These spikes manifest themselves as singular events localized in space and time; i.e. only a single pixel is hit at a time, and this event can not be noticed in the next measured image.

The transient pixel flagging algorithm for radiance data will work on partially processed earthshine data. The algorithm shall be executed after the slit irregularity correction and before the radiance sensitivity conversion. The algorithm requires radiance data and the associated noise as input. Both have to be expressed as electrons per second to allow measurements with different exposure times to be mixed. Additional input data like the thresholds have to be taken from the OPF data. The algorithm will work in a pipeline environment as long as the previous CCD readout is remembered. This step is necessary to normalize each frame.

## 1.1 Algorithm description

The algorithm is based on comparison of a measured image (frame) to its predecessor image by dividing with that image. Because radiance measurements are not expected to vary widely on short time-scales, this ratio will be close to unity. An additional smoothing can be applied per row in the column (i.e. the wavelength) direction. This holds because a change in ground-scene over time will in general manifest itself as a smooth change of the measured spectrum. If each row of the image is divided by a smooth version of that same row, the aforementioned fraction will be even closer to unity. The smooth version of the row under inspection is generated using a running median filter (like a boxcar filter) with a variable width. The same approach can be used to apply additional smoothing per column in the row direction. The algorithm does not flag the pixel if the signal-to-noise-ratio is too low. This in order to prevent nearly all pixels with low radiances to be flagged; the minimum required SNR is a tuneable parameter. Based on empirical observations, thresholds can now be defined that can distinguish between a ‘true’ measured change and a single radiation event. These thresholds are defined in the operational parameter file (OPF).

### 1.1.1 Applicable area

The transient flagging algorithm for radiance data should only work on the optic regions of the two CCD channels, i.e. the UV1, UV2 and VIS sub-channels.

### 1.1.2 Mixed orbits

Currently orbits are not foreseen that contain a combination of global measurements and measurements with another binning factor like spatial zoom, super zoom or spectral zoom. However the algorithm should check if the current frame is of the same type of the previous frame to ensure that the division is valid. If this is not the case, the sequence should be considered to restart, i.e. no pixels are flagged, the frame is stored, and the process should proceed to the next frame.

### 1.1.3 Flagged data handling

Pixels that have one of the following flags raised should be ignored by the transient detection algorithm:

- **Dead Pixel**
- **Missing Pixel**
- **Processing Error**

### 1.1.4 Properties of the median filter

- The median filter works like a boxcar average, with the difference that the filtered value are not calculated by the mean, but by the median.
- The width of the median filter is the amount of pixels that are included in the calculation of the median value.
- If the width is an odd number, the median value is the middle number.
- If the width is an even number, the median value is the mean of the two middle numbers.
- At the edges, the filter should fill all values from the edge to the width with the median value of the width. Example: if the width equals five than all five values at the edge are filled with the median value of these five elements; note that all these five elements are same. This feature is required to ensure that spikes at the edges can be found.
- If the OPF median width equals zero or one do nothing, i.e., skip smoothing and flagging in that direction.
- If the OPF median width is less than zero take the median of the whole array
- If the OPF median width is greater than one then smooth the row data using the specified width

### 1.1.5 Computational sequence

1. Determine whether the current frame is the first radiance frame in the current file. If this is true then store this frame in a buffer, raise no transient flags and proceed to the next frame.
2. Check if the previous frame has the same number of rows, number of columns and binning factor as the current frame. If this is not true then store the current frame in a buffer, raise no transient flags and proceed to the next frame.
3. Divide the current frame by the previous frame which is stored in the buffer. Prevent division by zero by padding with ones (1.0) if necessary. Also pad all pixels for which one of the flags specified in section 1.1.3 is raised.

4. Loop over all rows in the frame. Use a median filter to calculate a smooth version of each row of data from step 3. The width of this filter varies per optic region and is set in the OPF.
5. Divide the current row by the smoothed version of the row. Check for division by zero; if necessary pad with ones (1.0)
6. Loop over all pixels in the current row. For each pixel in this row determine whether the value calculated in step 5 minus one exceeds the allowed spike threshold as set by the OPF, and store this flag locally.
7. Repeat from step 4 for all rows.
8. Loop over all columns in the frame. Use a median filter to calculate a smooth version of each column of data as calculated in step 3. The width of this filter varies per optic region and is set in the OPF.
9. Divide the current column by the smoothed version of the column. Check for division by zero; if necessary pad with ones (1.0)
10. Loop over all pixels in the current column. For each pixel in this column determine whether the value calculated in step 9 minus one exceeds the allowed spike threshold as set by the OPF, and store this value as another flag.
11. Repeat from step 8 for all columns.
12. For all pixels  check if the signal-to-noise-ratio (SNR) of exceeds the SNR-threshold as set by the OPF. In case the SNR cannot be calculated due to flagging, then calculate no transient flag.
13. For each pixel check whether the flag from step 6 or 10 was raised (OR function).
14. Raise the transient flag for each pixel that has passed the conditions calculated in step 10 and 11 (AND function).
15. Store the current frame in the buffer in order to normalize the next frame.

### 1.1.6 Equations

$$normalized\_signal(i,j,m) = \frac{signal(i,j,m,f)}{signal(i,j,m,f-1)}$$

$$median\_filtered\_col(i,j,m) = MEDIAN(normalized\_signal(all,j,m), opf\_median\_width\_col(m))$$

$$median\_filtered\_row(i,j,m) = MEDIAN(normalized\_signal(i,all,m), opf\_median\_width\_row(m))$$

$$spike\_level\_col(i,j,m) = \frac{normalized\_signal(i,j,m)}{median\_filtered\_col(i,j,m)} - 1$$

$$spike\_level\_row(i,j,m) = \frac{normalized\_signal(i,j,m)}{median\_filtered\_row(i,j,m)} - 1$$

$$snr(i,j,m) = \frac{signal(i,j,m)}{noise(i,j,m)}$$

$$snr\_valid(i,j,m) = snr(i,j,m) \geq opf\_snr\_threshold(m)$$

$$spike\_flag\_row(i,j,m) = spike\_level\_row(i,j,m) \geq opf\_spike\_threshold\_row(m)$$

$$spike\_flag\_col(i,j,m) = spike\_level\_col(i,j,m) \geq opf\_spike\_threshold\_col(m)$$

$$spike\_flag(i,j,m) = spike\_flag\_row(i,j,m) \vee spike\_flag\_col(i,j,m)$$

$$transient\_flag(i,j,m) = snr\_valid\_flag(i,j,m) \wedge spike\_flag(i,j,m)$$

### 1.1.7 Definition of variables

In table 1 to 3, the description of the variables are given. The following abbreviations are used in the tables. Column C describes the characteristic of the variable (I=Input, L=Local, O=Output). Column D describes the type (df=float, ui=unsigned integer ). Column U describes the dimension (dl=dimensionless).

| variables | descriptive name | C | D | U | range | reference |
|---|---|---|---|---|---|---|
| signal(I,j,m,f) | Earths radiance measurement | I | df | e/s | | |
| noise(I,j,m) | Error in the measurement | I | df | e/s | | |
| opf_median_width_row(m) | Width of the median filter in the row dimension | I | ui | dl | | Operational parameter file |
| opf_median_width_col(m) | Width of the median filter in the column dimension | I | ui | dl | | Operational parameter file |
| opf_snr_threshold(m) | Signal-to-noise-ratio threshold | I | df | dl | | Operational parameter file |
| opf_spike_threshold_col(m) | Spike threshold in the column direction | I | df | dl | | Operational parameter file |
| opf_spike_threshold_row(m) | Spike threshold in row direction | I | df | dl | | Operational parameter file |

**Table 1: Input variables associated with the transient signal flagging algorithm**

| Variables | descriptive name | C | D | U | range |
|---|---|---|---|---|---|
| i | Column number in the optic region | L | ui | dl | |
| j | Row number in the optic region | L | ui | dl | |
| m | Optic region identifier | L | ui | dl | UV1, UV2, VIS |
| f | Sequential image identifier | L | ui | dl | |
| normalized_signal(I,j,m) | Normalized signal | L | df | dl | |
| median_filtered_row(I,j,m) | Median filtered row data | L | df | dl | |
| median_filtered_col(I,j,m) | Median filtered col data | L | df | dl | |
| spike_level_col(I,j,m) | Will be compared to the col thresholds | L | df | dl | |
| spike_leve_row(I,j,m) | Will be compared to the row thresholds | L | df | dl | |
| spike_flag_row(I,j,m) | Boolean | L | ui | dl | True, false |
| spike_flag_col(I,j,m) | Boolean | L | ui | dl | True, false |
| snr(I,j,m) | Signal-to-noise-ratio | L | df | dl | |
| snr_valid(I,j,m) | S/r high enough | L | ui | dl | True , false |

**Table 2: Local variables associated with the transient signal flagging algorithm**

| variables | descriptive name | C | D | U | range |
|---|---|---|---|---|---|
| Transient_flag(I,j,m) | Transient flag | O | ui | dl | True, false |

**Table 3: Output variables associated with the transient signal flagging algorithm**

## 1.2 OPF data for transient signal flagging of radiance data

The OPF data for transient pixel flagging of radiance data contains settings for each optic region. The three optic regions are denoted by UV1, UV2 and VIS. For each region, a threshold is given for the “spike level” and a threshold for the signal-to-noise-ratio. Furthermore, the width of the median filter for each optics region is specified. If the median width is set to zero then no spike detection in that direction is done. Currently the spike detection in the row direction is switched of; if switched on the width should be approximately 5. The values of these thresholds are given in the following tables:

| Column spike level | Row spike level | SNR threshold |
|---|---|---|
| 0.1 | 0.5 | 18.0 |

**Table 4: OPF filter thresholds for the UV1 channel**

| Column median filter width | Row median filter width |
|---|---|
| 11 | 0 |

**Table 5: OPF filter widths for the UV1 channel**

| Column spike level | Row spike level | SNR threshold |
|---|---|---|
| 0.1 | 1.0 | 20.0 |

**Table 6: OPF filter thresholds for the UV2 channel**

| Column median filter width | Row median filter width |
|---|---|
| 11 | 0 |

**Table 7: OPF filter widths for the UV2 channel**

| Column spike level | Row spike level | SNR threshold |
|---|---|---|
| 0.1 | 1.0 | 40.0 |

**Table 8: OPF filter thresholds for the VIS channel**

| Column median filter width | Row median filter width |
|---|---|
| 11 | 0 |

**Table 9: OPF filter widths for the VIS channel**

## 1.3 Comments on prototype code

The IDL prototype code is attached in Appendix A. This code ignores the pipeline environment of the actual data processor. It is only supplied to clarify the algorithm description and to be used as an example. IDL codes are overloaded to support vector data, so no explicit references to row or column numbers are needed. A star (*) used in the indexing of an array denotes that the operation works on all elements of the array in that specific dimension.

## 2 Test results and examples

Examples of two spikes are given in Figure 1 and Figure 2. The first example is that of an extremely strong spike, while the latter is of medium strength. Each figure is for a single frame and a full column, and contains four panels. The top panel shows the spike level value as calculated by the algorithm; this value is checked versus the threshold to identify spikes. The bottom panel shows the signal-to-noise-ratio of the measurement at that column; this has to exceed a certain level in order to flag is spike as a transient. The second and third panel show the radiance data on a normal and a logarithmic scale consecutively; both plots are given in pixel charge in units of electrons per second. The diamonds in the plots indicate the position of the spike.

Note that some spikes seem to be very small; inspection of the radiance data on a logarithmic scale clearly indicates that the spikes are well above the expected signal level. Furthermore, it becomes evident that more spikes are flagged at the higher column numbers. This is due to the fact that the signal values are lower, and thus that a small spike has a higher impact. The signal-to-noise-ratio check prevents too many pixel to be flagged at extremely low pixel fillings.

In Figure 3 the algorithm is demonstrated on a single frame (measurement) acquired during a passage of the Southern Atlantic Anomaly. The data shown is approximately half of the UV1 optic region. This part has very low radiances due to the short wavelengths; therefore, spikes are very well visible here. The figure contains two panels, the upper is the radiance data in electrons per second; many spikes can be seen clearly. The lower panel depicts the same data, but now without the pixels that the algorithm has identified as having a spike. The majority of the spikes are removed, with some exceptions at the border of the channel. This is due to the fact that the radiances are extremely low, too low for the algorithm to work properly. This is not considered a problem because this data is unreliable to start with.

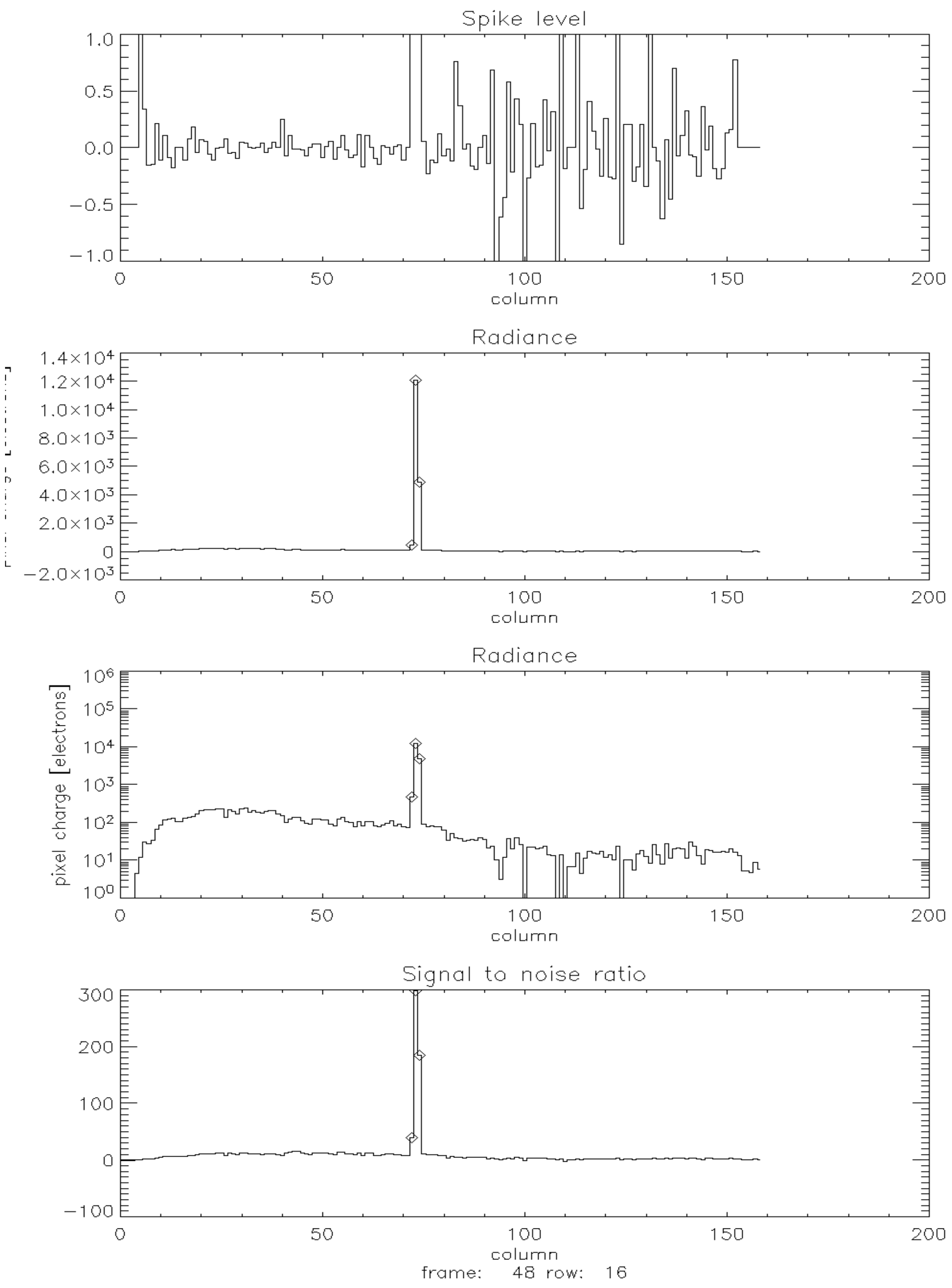


**Figure 1: Example of a very strong spike at column 71**

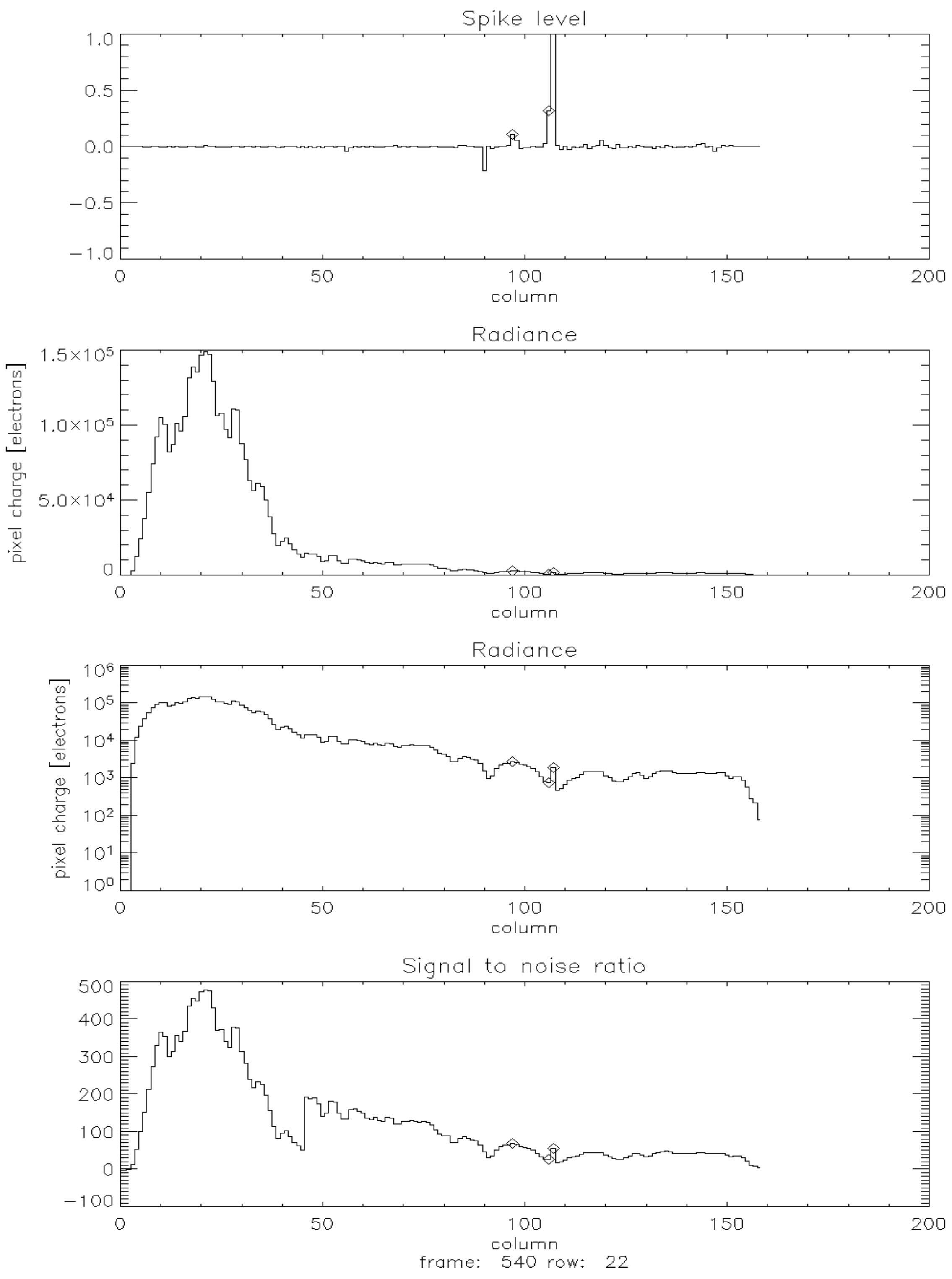
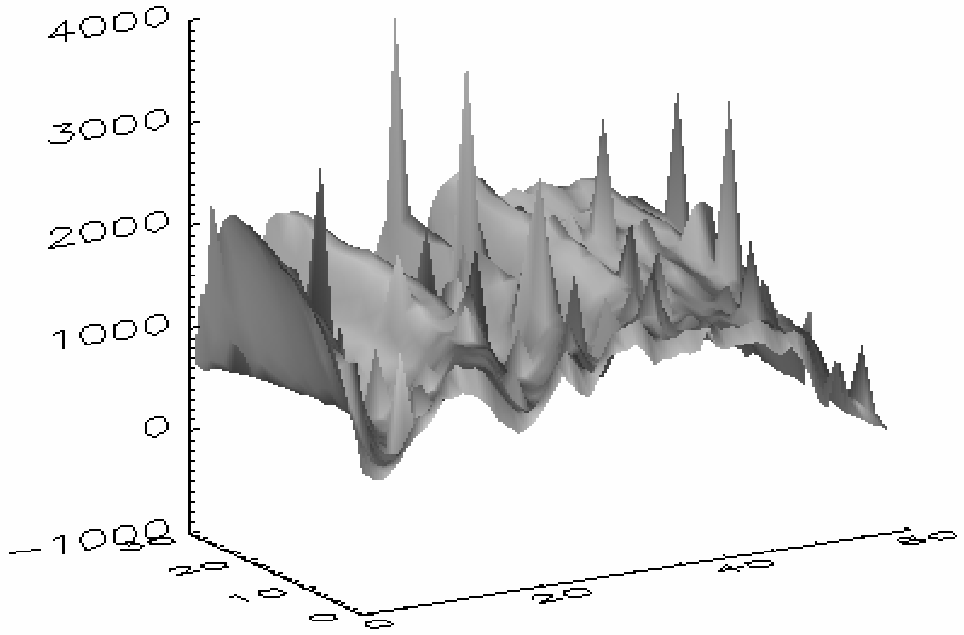


**Figure 2: Example of a medium strength spike at column 108 .**



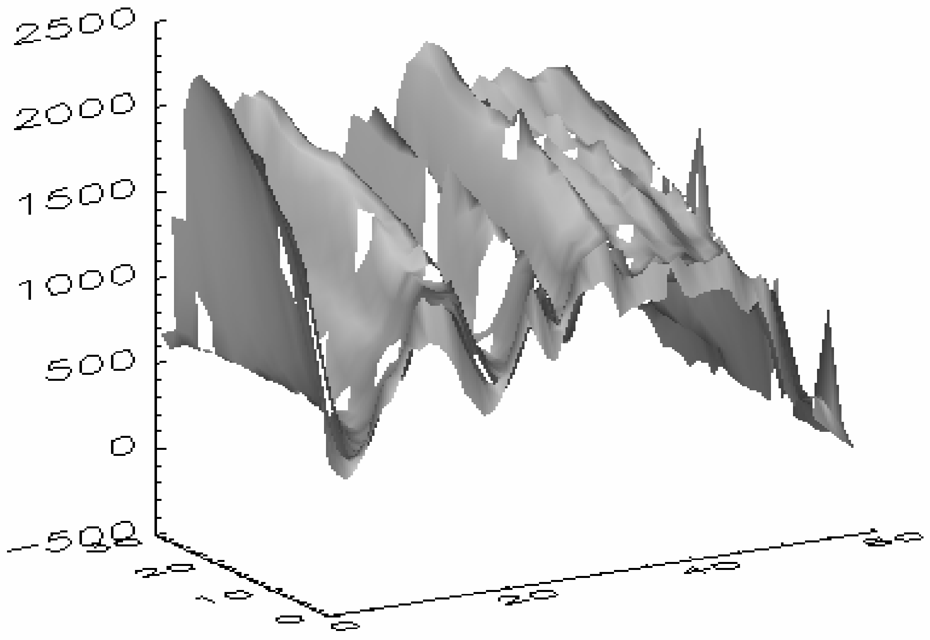


**Figure 3: Radiance in UV1 channel during crossing of the SAA. Top panel shows raw signal, bottom panel signal with spikes removed.**

# Appendix A IDL prototype source code

```
;-----
;
; SPYKER algorithm for radiance data
; detects spikes in processed radiance data, given in electrons,
; using sequential comparison for all frames, ignoring the first
; frame.
; returns an array with the same size of the input data containing
; flags for pixels with a spike
; these pixels should be excluded when the data is averaged onwards
;
; KNMI, 28-apr-2005 13:50
; version 1.2(radiance)
; Q. L. Kleipool
;
;-----

function Spyker , Data , Error , Settings , VALUES=RowValues


;
; Data is a 3 dimensional array containing raw data rames for a
; single optics region
; Error contains the corresponding noise values
; On exit the Values variable contains the 'cost function'
; this is only used as a diagnostic to tune the threshold parameters
;

ColumnCount = n_elements(Data[*,0,0])
RowCount    = n_elements(Data[0,*,0])
FrameCount  = n_elements(Data[0,0,*])


;
; and declare some variables
;

Normalised   = fltarr(ColumnCount,RowCount,FrameCount)
ColumnValues = fltarr(ColumnCount,RowCount,FrameCount)
RowValues    = fltarr(ColumnCount,RowCount,FrameCount)
ColumnMask   = bytarr(ColumnCount,RowCount,FrameCount)
RowMask      = bytarr(ColumnCount,RowCount,FrameCount)
SpikeMask    = bytarr(ColumnCount,RowCount,FrameCount)


;
```

```
; determine the valid Signal-to-Noise-Ratio flag
;

SNR_valid = (Data/Error) GT Settings.SnrThreshold

;
; divide each frame with the previous frame
; to bring all values close to one
; set the normalised values for the first frame to 1
;

Normalised[*,*,0] = 1.
for Frame=1,FrameCount-1 do begin
  Normalised[*,*,Frame] = Data[*,*,Frame] / Data[*,*,Frame-1]
endfor

;
; apply additional smoothing in the column direction
; this is the wavelength direction
; if the OPF median width equals zero do nothing
; if the OPF median width is less than zero take the median of the
; whole row
; if the OPF median width is greater than zero then smooth the row
; data using a media filter of the given width
;

for Frame=0,FrameCount-1 do begin
  for Row=0,RowCount-1 do begin
    ThisRow = reform(Normalised[*,Row,Frame])
    RowFit  = fltarr(ColumnCount) + 1.
    if Settings.ColumnWidth EQ -1 then RowFit = median(ThisRow)
    if Settings.ColumnWidth GT  1 then RowFit = median(ThisRow,Settings.ColumnWidth)
    ColumnValues[*,Row,Frame] = (ThisRow/RowFit) - 1
  endfor
endfor

;
; do the same in the row direction
;

for Frame=0,FrameCount-1 do begin
  for Col=0,ColumnCount-1 do begin
    ThisCol = reform(Normalised[Col,*,Frame])
    ColFit  = fltarr(RowCount) + 1.
    if Settings.RowWidth EQ -1 then ColFit = median(ThisCol)
    if Settings.RowWidth GT  1 then ColFit = median(ThisCol,Settings.RowWidth)
    RowValues[Col,*,Frame] = (ThisCol/ColFit) - 1
  endfor
endfor

;
; check where the thresholds are exceeded for all data elements
;

if Settings.ColumnWidth NE 0 then ColumnMask = (ColumnValues GT
Settings.ColumnThreshold)
if Settings.RowWidth    NE 0 then RowMask    = (RowValues    GT
Settings.RowThreshold)

print,'flagged in column direction: ',long(total(ColumnMask AND SNR_valid))
print,'flagged in row direction    : ',long(total(RowMask    AND SNR_valid))

;
; a spike is now defined when it is flagged in either the row analysis or the column
; analysis
; in the final mask only pixels are flagged as spiked when they have a sufficient
; signal to noise ratio
;

SpikeMask = ColumnMask OR RowMask
FinalMask = SpikeMask AND SNR_valid

return,FinalMask
end

;-----
```